CS 188 Discussion 5: Reinforcement Learning (For real this time)

TA: Sherdil Niyaz

Administrivia

* Midterm Thursday!!!
* Project due 10/14

Recall: MDP's

* Definition of an MDP:
* Set of states: $s_1, s_2, s_3, \ldots$
* Reward for taking an action between states: $R(s, a, s')$
* Transition function: $P(s'|s, a)$
* And, of course, actions that we can take at each state!

Recall: MDP's

* Definition of an MDP:
* Set of states: $s_1, s_2, s_3, \ldots$
* Reward for taking an action between states: $\cancel{R(s, a, s')}$
* Transition function: $\cancel{P(s'|s, a)}$
* And, of course, actions that we can take at each state!

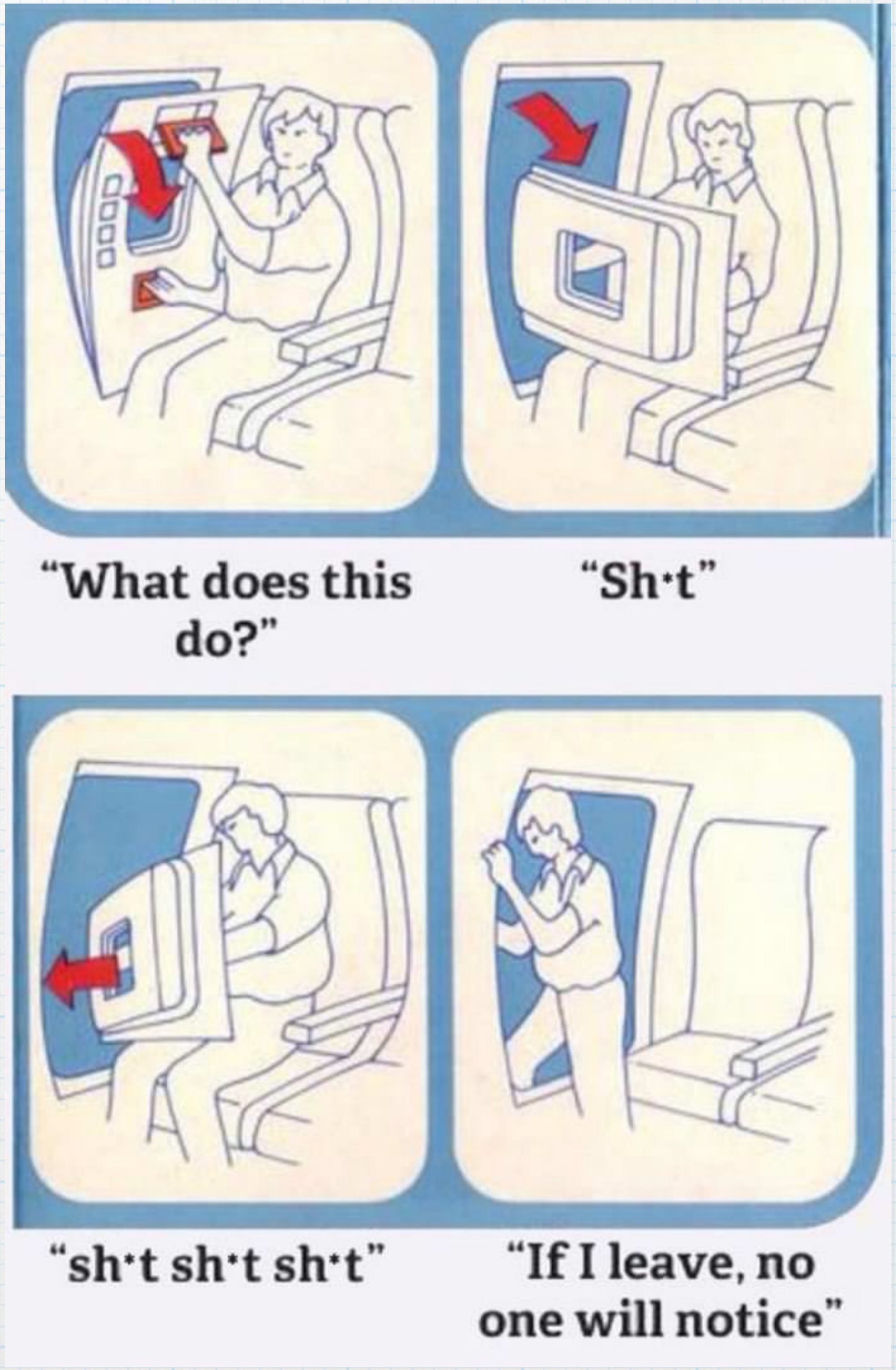
What if you don't know these? :(

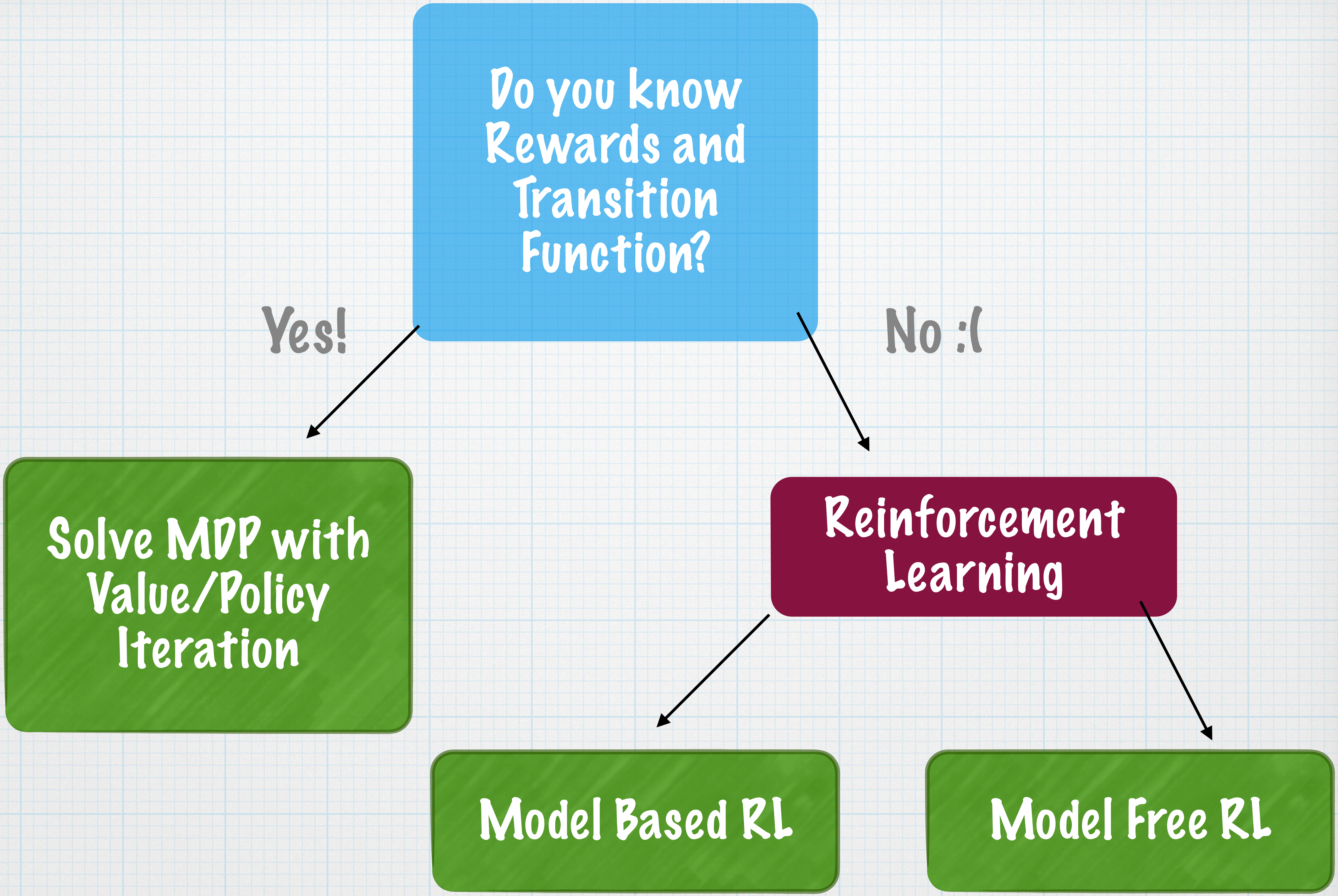
Reinforcement Learning !

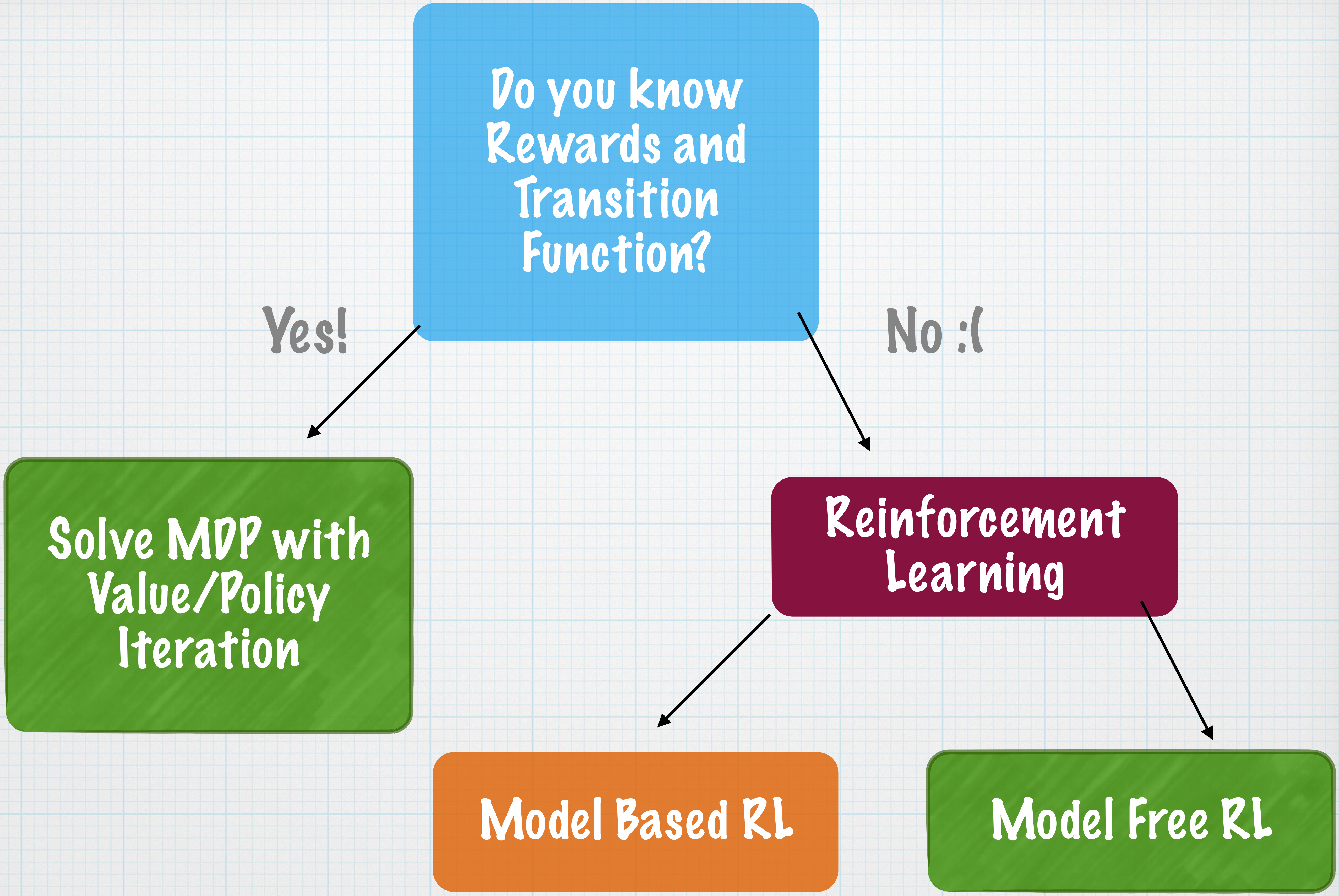
* Basic Idea: we don't know now what our actions can do, or even what the reward of those actions will be!
* So, let's just try some stuff, and try to find out what we should do along the way!

Downside...

* While you're figuring stuff out, you have to make some **really bad** decisions to figure out what's actually bad....

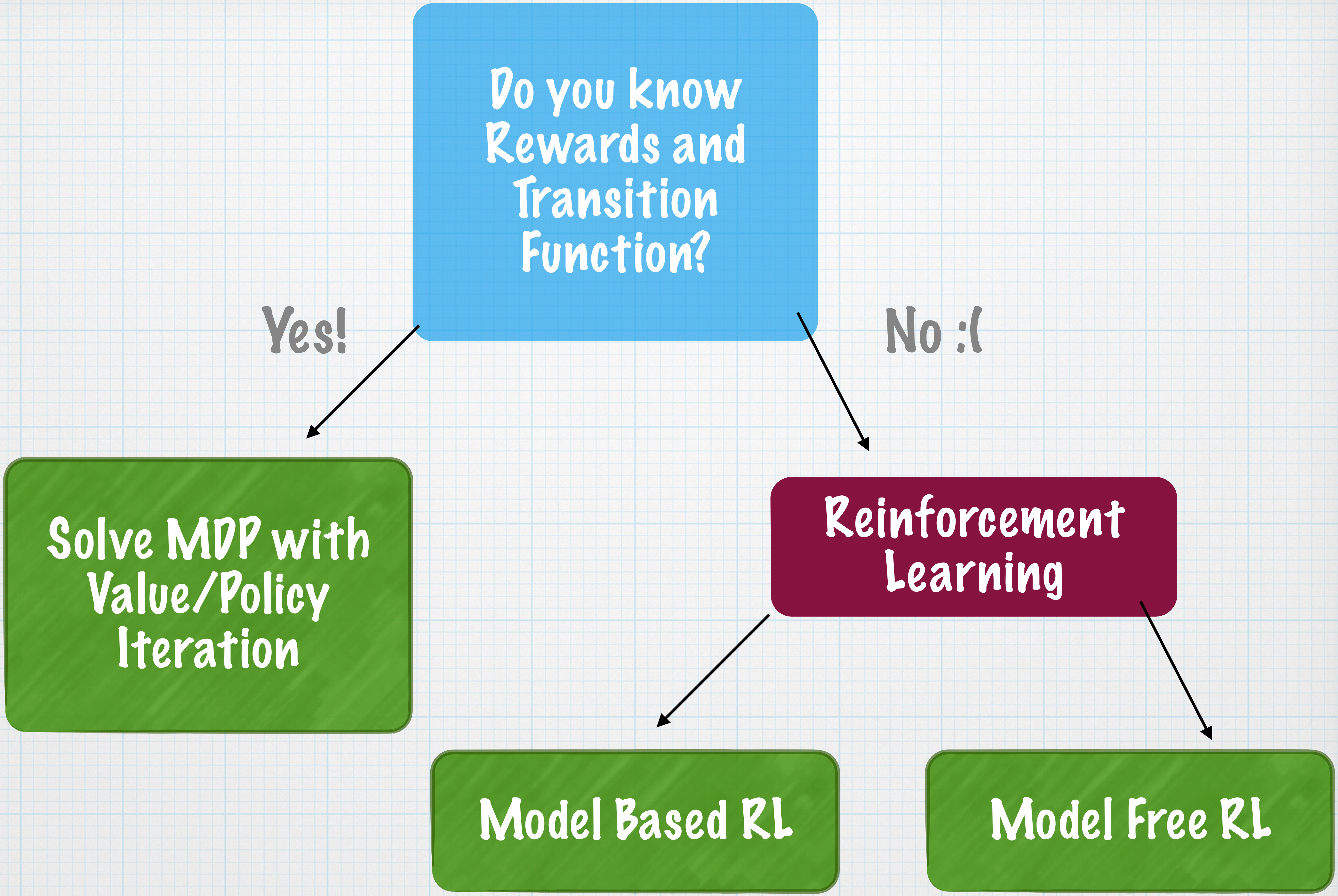



Do you know Rewards and Transition Function?
Yes!
No :(
Solve MDP with Value/Policy Iteration
Reinforcement Learning
Model Based RL
Model Free RL

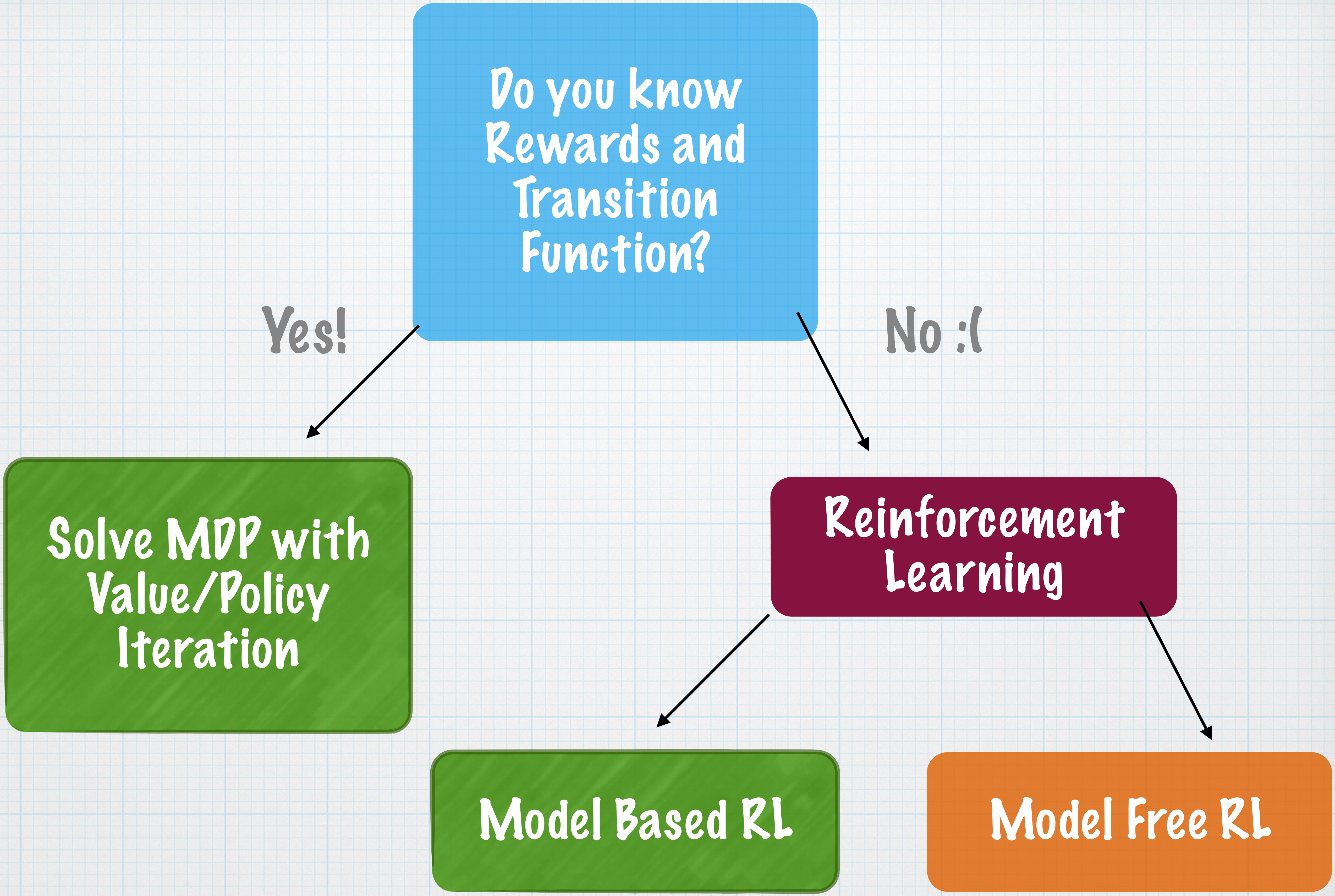

Do you know Rewards and Transition Function?
Yes!
No :(
Solve MDP with Value/Policy Iteration
Reinforcement Learning
Model Based RL
Model Free RL

Model Based RL

* Let's try to estimate -> $P(s'|s, a)$
* Let's Observe each -> $R(s, a, s')$
* Do this enough, and you'll get the correct values for each!
* Now **solve like regular MDP.**



Do you know Rewards and Transition Function?
Yes!
No :(
Solve MDP with Value/Policy Iteration
Reinforcement Learning
Model Based RL
Model Free RL


Do you know Rewards and Transition Function?
Yes!
No :(
Solve MDP with Value/Policy Iteration
Reinforcement Learning
Model Based RL
Model Free RL

Model Free RL

* Don't try to find "structure" in the problem.
* Aka, don't care about discovering reward functions and transition function.
* Instead, just take an action and **see what happens**. Learn what actions are good this way, even if you don't know **why** they're good!

Q-Learning

* Very common approach to Model Free Learning. Learns what taking an action **a** from a state **s** does.
* We learn what's good, but not necessarily **why** those actions are good.
* Scary Equation hopefully makes more sense now.

Scary Equation

$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

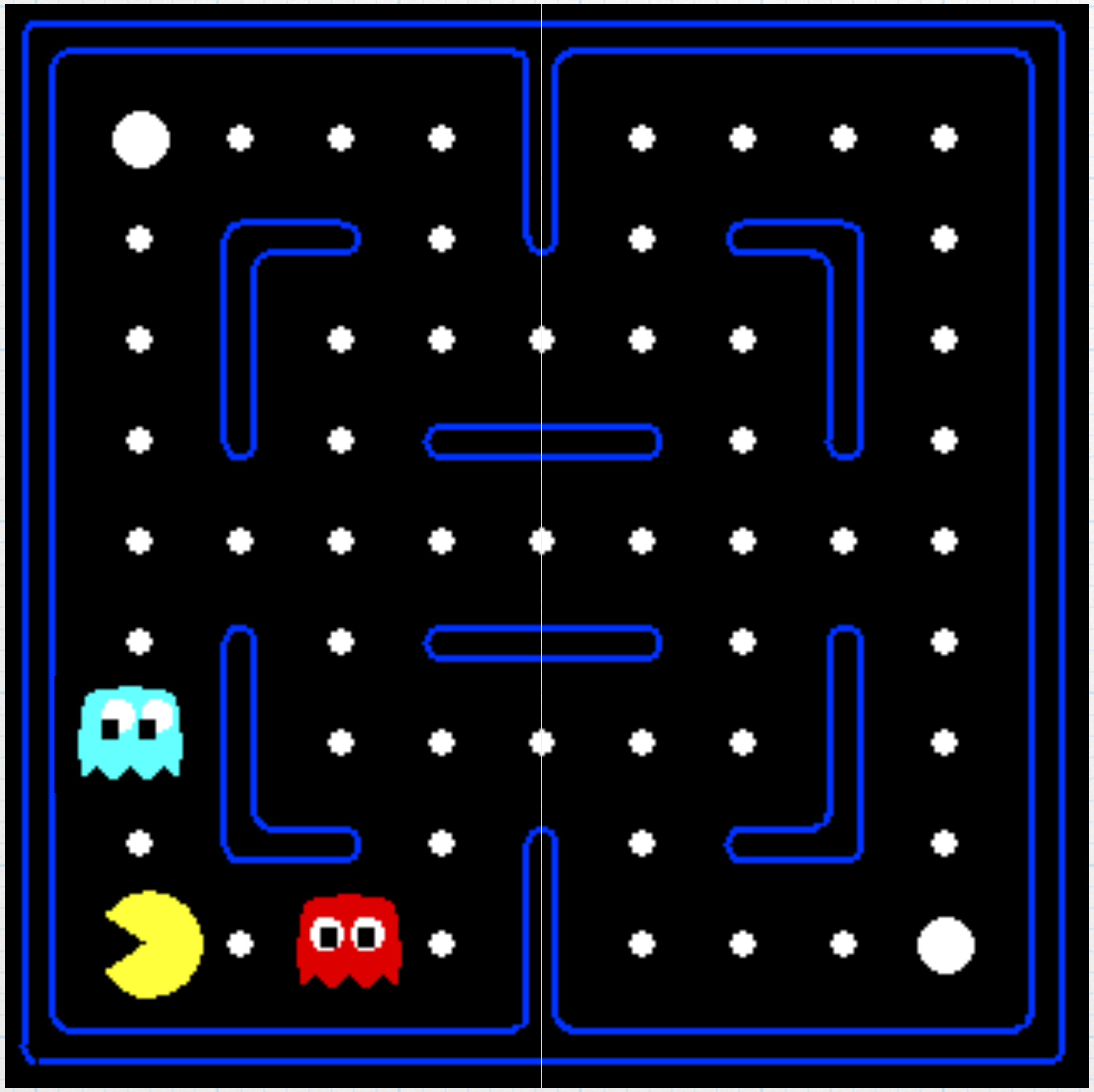
$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + (\alpha) \, [sample]$$

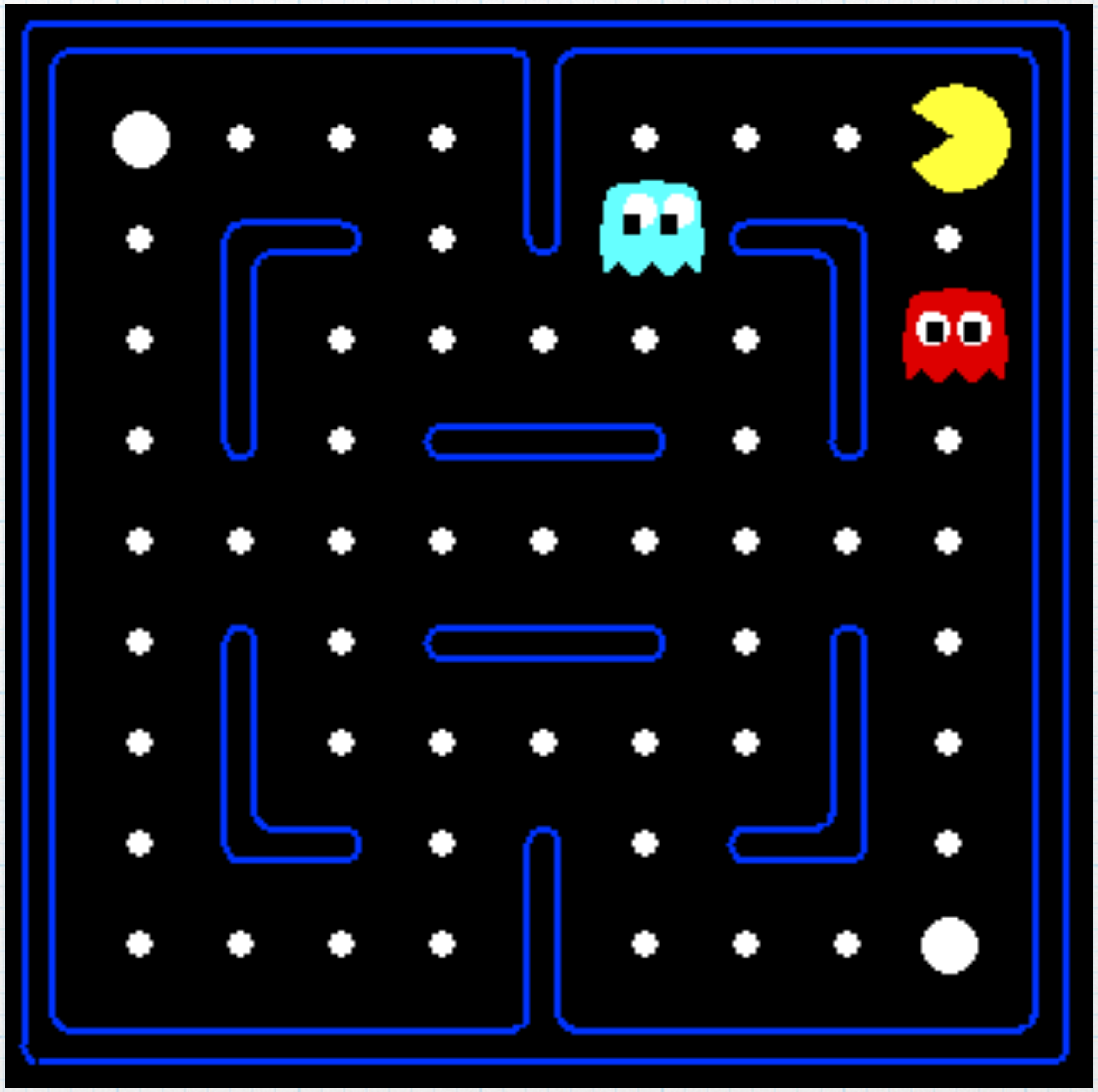
This is a **moving average** that incorporate both what we've seen as well as our old experiences!

Two problems with "normal" Q-Learning

* Each state + action pair Q(s,a) has its own value. We have to keep track of a **huge** number of Q(s,a) values!
* We also don't know **anything** about states we've never seen, even if they're very similar to states we have seen!

Shouldn't these two states have similar values???





Solution: Feature Based Q-Learning!

* Each state + action pair now is now represented as a vector or features
* Value Q(s,a) is **weighted sum** of those features!

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

Solution: Feature Based Q-Learning!

* You have to pick these features! Just like Project 2.

* Don't bother learning Q(s,a) directly- just **learn the weights!**

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

(We have one set of weights we use for all states!)

Normal Q-Learning Update

$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha)\,[sample]$$

Now re-write it!

(This is **sample** from above!)

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s, a)$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,[\text{difference}]$$

Normal Q-Learning Update

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s, a)$$

What actually happened (sample)

What we expected

$$Q(s, a) \leftarrow Q(s, a) + \alpha \, [\text{difference}]$$

(The bigger the gap, the more we update!)

Featurized Q-Learning Update

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a') \right] - Q(s, a)$$

$$w_i \leftarrow w_i + \alpha \, [\text{difference}] \, f_i(s, a)$$

Feature-based version: learn the **weights** instead of learning Q(s, a) directly!

Featurized Q-Learning Update

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s, a)$$

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s, a)$$

Notice: the more a feature contributed, the more of an update its weight gets!

Remember:

In learning the weights, you learn Q(s, a) values!

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$